

Annotating Derivations: A New Evaluation Strategy and Dataset for Algebra Word Problems

Shyam Upadhyay¹ Ming-Wei Chang²

¹ University of Illinois at Urbana-Champaign, IL, USA

² Microsoft Research, Redmond, PA, USA

upadhya3@illinois.edu

minchang@microsoft.com

Abstract

We propose a new evaluation for automatic solvers for algebra word problems, which can identify reasoning mistakes that existing evaluations overlook. Our proposal is to use *derivations* for evaluations, which reflect the reasoning process of the solver by explaining how the equation system was constructed. We accomplish this by developing an algorithm for checking the equivalence between two derivations, and showing how derivation annotations can be semi-automatically added to existing datasets. To make our experiments more comprehensive, we also annotated DRAW-1K, a new dataset of 1000 general algebra word problems. In total, our experiments span over 2300 algebra word problems. We found that the annotated derivation enable a superior evaluation of automatic solvers than previously used metrics.

1 Introduction

Automatically solving math reasoning problems is a long-pursued goal of AI (Bobrow, 1964; Hegarty et al., 1995). Recent work (Kushman et al., 2014; Shi et al., 2015; Koncel-Kedziorski et al., 2015) has focused on developing solvers for *algebra word problems*, such as the one shown in Figure 1. This task can be viewed as a semantic parsing problem, where the parser (the solver) reads a natural language text (the word problem), performs reasoning, outputs an executable representation (the *equation system*), and generates the response (the *solution*) using the representation.

Developing an efficient solver for word problems can open several new avenues, especially for online education systems. In addition, as solving

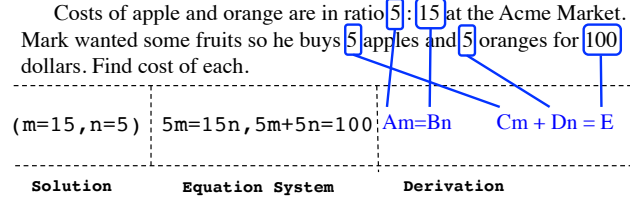


Figure 1: A algebra word problem with its solution, equation system and derivation. We show that evaluation on derivation is a more accurate assessment of reasoning, than evaluating with solution or equation system.

word problems requires the ability to understand natural languages and analyze the relationships between numerical values, it provides a good test bed for evaluating the progress of artificial intelligence techniques, a direction advocated by Clark and Etzioni (2016).

In both application scenarios, the ability for the automatic solver to generate the right solutions is often not the right evaluation criteria. It is more important that the automatic solver can show *how* it obtained the solutions, e.g. the derivation. A derivation consists of a *template* ($\{Am = Bn, Cm + Dn = E\}$ in Figure 1) and *alignments* of numbers in the text to its coefficients (blue edges), which together describe the construction of an equation system, thereby reflecting the reasoning process of the solver.

Surprisingly, to the best of our knowledge, no prior work evaluates the quality of the solvers by evaluating derivations directly. Current studies mainly use two strategies for evaluating the word problem solvers. The most popular strategy is to use *solution accuracy* (whether the solution was correct or not) (Kushman et al., 2014; Hosseini et al., 2014; Shi et al., 2015; Koncel-Kedziorski et al., 2015; Zhou et al., 2015; Huang et al., 2016), as this is the easiest measurement to implement. The other strategy is to calculate the *equation ac-*

curacy¹ (Kushman et al., 2014; Zhou et al., 2015), which can be considered as comparing derivations approximately (as there is no gold labeled derivation before this work). Unfortunately, as we will point out later, these two evaluation strategies are not precise enough and can be significantly different from evaluating on derivations directly. Equation accuracy, due to its approximation natural, can even be misleading by giving a higher score to a worse solver.

While evaluating on derivation is desirable, there are two reasons prevent existing studies from adopting derivation evaluation. First, comparing derivations is not straightforward. Two different templates can end up being semantically equivalent after algebra manipulation. Also, the evaluation procedure also needs to examine if the alignments between textual numbers and coefficients of two derivations agree with each other. Second, there is no gold annotation for derivations, as existing data sets only often contain equation systems and solutions.

In this paper, we address these concerns and propose to evaluate the solvers using *derivation accuracy*. We conduct an extensive study on comparing different evaluation strategies. The contributions of our papers is as follows:

- We give a formal definition on the equivalence between two derivations, and use this definition to develop a novel algorithm for computing derivation accuracy.
- We show how derivation annotations can be semi-automatically generated by transforming existing datasets, which only contain word problems and their equation systems. We labeled over 2300 word algebra problems with detailed derivation annotations.
- We point out that evaluating using derivations is more precise, and results on derivation accuracy can be significantly different from results on solution accuracy and equation accuracy.

We publicly release all problems with detailed derivation annotations, from existing datasets and

¹In fact, equation accuracy is not a well defined term, as it is not clear how to compare two equation systems. Different papers use different procedures for calculating the equation accuracy (Kushman et al., 2014; Zhou et al., 2015). In this paper, we mainly adopt the version proposed by (Kushman et al., 2014), where they inferred the template from the equations system without using the gold derivation.

Word Problem	x	Alice’s and Bob’s speed are in the ratio 5 : 15, and they are 100 miles apart. They ran towards each other and met after five hours. Find their speeds.
Textual Numbers	$\mathcal{Q}(x)$	{5, 15, 100, five}
Equation System	y	$5m = 15n, 5m + 5n = 100$
Solution		$m = 15, n = 5$
Template	T	$Am = Bn, Cm + Cn = D$
Coefficients	$\mathcal{C}(T)$	A, B, C, D
Alignments	A	{5 \rightarrow A, 15 \rightarrow B, five \rightarrow C, 100 \rightarrow D}
Derivation	z	(T, A)

Table 1: The symbols we used in the paper. Our proposed annotations are shown in **bold**. Note that without derivation it is impossible to distinguish between “5” and “five” just using y above.

DRAW-1K, a new dataset we introduce in this paper. All of them can be downloaded at <https://aka.ms/datadraw>.

2 Evaluating Derivations

Preliminaries We introduce our notation using the example word problem in Table 1. We denote a word problem by x and an *equation system* by y . A *textual number* is a span of text in x that indicates a numeric quantity. $\mathcal{Q}(x)$ is the set of all textual numbers.²

A *template* T is a *parameterized* semantic parse, whose parameters are coefficients $\mathcal{C}(T) = \{c_i\}_{i=1}^k$, where each coefficient c_i can align to a single textual number. An *alignment* is a pairing (q, c) between a textual number $q \in \mathcal{Q}(x)$ and a coefficient $c \in \mathcal{C}(T)$.

A set of alignments A and a template T together identify a *derivation* $z = (T, A)$ of an equation system. We use (A, B, C, \dots) to represents coefficients and (m, n, \dots) to represent the unknown variables in the templates. We assume there exists a routine `Solve`(y) that provides the solution of an equation system. We use a Gaussian elimination solver for our `Solve` routine.

Derivation Equivalence We define two derivations (T_1, A_1) and (T_2, A_2) to be equal if the corresponding templates T_1, T_2 and alignments A_1, A_2 are equivalent.

Intuitively, two templates T_1, T_2 are equivalent if they can generate the same space of equation systems – i.e., for every assignment of values to slots of T_1 , there exists an assignment of values to

²We use the quantity normalizer in Stanford CoreNLP (Manning et al., 2014) to identify textual numbers.

Algorithm 1 Judging Template Equivalence.

Input: Two templates T_1 and T_2 with coefficients $\mathcal{C}(T_1)$ and $\mathcal{C}(T_2)$ respectively, such that $|\mathcal{C}(T_1)| = |\mathcal{C}(T_2)|$; R : Rounds

```
1:  $\Gamma \leftarrow \emptyset$ 
2: for each 1-to-1 mapping  $\gamma : \mathcal{C}(T_1) \rightarrow \mathcal{C}(T_2)$  do
3:    $\text{match} \leftarrow \text{True}$ 
4:   for  $t = 1 \dots R$  do
5:     Generate random vector  $\mathbf{v}$ 
6:      $A_1 \leftarrow \{(\mathbf{v}_i \rightarrow c_i)\}, A_2 \leftarrow \{(\mathbf{v}_i \rightarrow \gamma(c_i))\}$ 
7:     if  $\text{Solve}(T_1, A_1) \neq \text{Solve}(T_2, A_2)$  then
8:        $\text{match} \leftarrow \text{False}$ ; break
9:   end if
10:  end for
11:  if  $\text{match}$  then  $\Gamma \leftarrow \Gamma \cup \{\gamma\}$ 
12: end for
13: return  $\Gamma$   $\triangleright \Gamma \neq \emptyset$  iff the templates are equivalent
```

slots of T_2 such that they generate the same equation systems. Similarly, two alignments A_1 and A_2 are equivalent if corresponding slots from each template align to the same textual number. In the following, we carefully explain how template and alignment equivalence are determined.

Template Equivalence Here we propose an algorithm that can automatically detect the equivalence between two templates. The algorithm will also find out all the valid mappings of coefficients between two templates. We use Algorithm 1 to identify equivalent templates, using the fact that under appropriate renaming of coefficients, two equivalent templates will generate equations which have the same solutions, for all possible coefficient assignments.

For two templates T_1 and T_2 , with the same number of coefficients $|\mathcal{C}(T_1)| = |\mathcal{C}(T_2)|$, we represent a choice of renaming coefficients by γ , a 1-to-1 mapping from $\mathcal{C}(T_1)$ to $\mathcal{C}(T_2)$. The two templates are equivalent if there is a γ such that solutions of the equations identified by T_1 and T_2 are same, for all possible coefficient assignments. The algorithm exhaustively tries all possible renaming of coefficients, checking if the solutions of the equation systems generated from a random assignment match exactly. It reports equivalence if for a renaming γ , the solutions match for $R = 10$ such random assignments. Despite this approximation, Algorithm 1 is quite effective, as we manually examine the annotation and did not find out any false positive case produced by the algorithm.

Alignment Equivalence After running the template equivalence algorithm, we have obtained every mapping γ in Γ under which the templates were equivalent. Recall that γ identifies corre-

sponding slots c and $\gamma(c)$ in T_1 and T_2 respectively. Two alignments A_1 and A_2 are equivalent if corresponding slots (according to γ) from each template align to the same textual number. More formally, if we find a mapping γ such that for each alignment (q, c) in A_1 there is alignment $(q, \gamma(c))$ in A_2 , then the alignments are equivalent.

We declare derivations (T_1, A_1) and (T_2, A_2) to be equivalent after checking both their template and alignment equivalent of each other.

3 Annotating Derivations

As none of the existing benchmarks contain derivation annotations, we decided to augment existing datasets with these annotations. We also created DRAW-1K, a new dataset of 1000 general algebra word problems to make our study more comprehensive.

Annotating gold derivations from scratch for all problems is time consuming. However, all word problems do not require manual annotation, as sometimes a number appearing in an equation may be uniquely aligned to a textual number. We first identify word problems which have ambiguity for alignments – problems which have multiple textual numbers with the same value which appear in the equation system. We only semi-automatically annotate problems which involve such alignment ambiguities.³ For the remaining problems, the annotations are generated fully-automatically.

Our approach involves two stages, where the first stage finds the correct template and alignments, while the second stage prunes equivalent pairs of templates.

In the first stage, each number appearing in \mathbf{y} is replaced with a coefficient c if a textual number $q \in \mathcal{Q}(\mathbf{x})$ with the same value is found in \mathbf{x} . The alignment (q, c) is also added to the alignment set A_i . If there are multiple valid choices of q , we choose the first one. All numbers in \mathbf{y} with the same value are assumed to map to the same coefficient. The equation system \mathbf{y} , whose numbers have now been replaced with coefficients, becomes the template T_i . Then, we ask a human annotator to correct the alignment for the numbers which cause ambiguity, that is, numbers in the equation which can align to multiple textual numbers. Once the alignments are corrected by the annotator, the templates are re-generated to reflect the change.

³However, annotations for all problems are verified later.

Dataset	DRAW-1K	ALG-514	DOLPHIN-L
# problems	1000	514	832
vocab.	2.21k	1.83k	0.33k

Table 2: Statistics of DRAW-1K, ALG-514, and DOLPHIN-L. Vocabulary was computed after removing all numbers in the text, as we are only looking for textual variation.

In the second stage, we use Algorithm 1 to automatically reconcile the templates in the derivations by finding pairs of equivalent templates, and replacing one with the other in the dataset. This remove redundancy in the annotations – if two templates are equivalent, it is desirable to only use one of them to in the annotations. We reconcile the templates to reveal the true complexity of the dataset, as we found that that the number of templates can be significantly overestimated by over 30% without properly reconciliation.

4 Experimental Setup

In the following, we describe the datasets used in our experiments. The dataset ALG-514 was introduced in (Kushman et al., 2014). It consists problems crawled from `algebra.com`, ranging over a variety of narrative scenarios (distance-speed, object counting, simple interest, etc.).

DOLPHIN-L is the linear-T2 subset of the DOLPHIN dataset (Shi et al., 2015). DOLPHIN focuses on a special kind of algebra word problems, e.g. *number word problems*, which describe mathematical relationships directly in the text.

We introduce *Diverse Algebra Word* (DRAW-1K), a new dataset, in this paper. It consists 1000 word problems crawled from `algebra.com`. Details on the dataset creation can be found in the appendix.

We use 5-fold cross validation splits provided by the authors for DOLPHIN-L and ALG-514. We randomly split DRAW-1K into train, dev and test splits with 600, 200, 200 problems respectively.

Statistics comparing the 3 datasets is shown in Table 2. Of all the datasets, DRAW-1K is the largest dataset on general algebra word problems.

4.1 Evaluation

We compare the following evaluation metrics.

Derivation Accuracy A derivation is correct if the predicted derivation’s template is equivalent to the reference derivation’s template, and the set of aligned textual numbers match those of the reference derivation, as we defined it in Section 2.

Solution Accuracy Following previous work (Kushman et al., 2014), we compute solution accuracy by checking if each number in the reference solution appears in the generated solution (disregarding order).

Equation Accuracy An approximation of derivation accuracy used in Kushman et al. (2014). A reference derivation \tilde{z} is randomly chosen from the set of derivations which construct the gold y from x . Derivation accuracy is computed against this reference derivation. Without human correction, the reference derivation can be incorrect, as several derivations may lead to the same y .

5 Experiments

We aim to answer the following questions in our experiments:

Given that we want to evaluate the reasoning ability for word problem solvers, are solution accuracy and equation accuracy good approximation of derivation accuracy?

To answer our questions, we need to construct two solvers where one of it has clear advantages, and see if the evaluation metrics can reflect this advantage. To construct such cases, we use the following settings, which are designed to show the value of each stage of our transformation in § 3. We evaluate the solver using the metrics described in § 4.1.

TE (TRAIN ON EQUATION) Only the (x, y) pairs are provided as supervision. Similar to (Kushman et al., 2014; Zhou et al., 2015), we use an approximation algorithm to find a derivation that agrees with the equation system and the solutions. This setting represents the supervision used in existing approaches (our baseline).

TD (TRAIN ON DERIVATION) (x, z) pairs obtained by the full transformation are used as supervision.

It should be obvious that TD setting is a more informative supervision strategy than the TE setting, as it provides labeled semantic parses (i.e. derivations) instead of only question-answer pair. In fact, as we will show it later, TD is better than TE on both solution and derivation accuracy on all cases.

Main Results We investigate if solution accuracy and equation accuracy are good approximation for derivation accuracy using the models in Table 3. We want to determine to what degree each evaluation metric reflects the superiority of TD over TE.

We can note that solution accuracy always exceeds derivation accuracy, as a solver can sometimes get the right solutions even with the wrong derivation. Also note that solution accuracy can sometimes hide the true improvement in reasoning ability. For instance, solution accuracy only changes by 2.4 on Dolphin when comparing TE and TD, whereas derivation accuracy changes by 10.7 points, clearly showing TD is more informative supervision setting.

On the other hand, equation accuracy (the approximated version of derivation accuracy) has several issues. In cases like DRAW-1K, equation accuracy cannot determine which model is better and assigns them the same score. Furthermore, it often considers TD to be a worse setting than TE, as evident from decrease in the scores. For instance, on DOLPHIN-L (TE) to DOLPHIN-L (TD), while both solution accuracy and derivation accuracy improve, equation accuracy *worsens* by 13.3 pts. Recall that equation accuracy attempts to approximate derivation accuracy by choosing a reference derivation, which can possibly be incorrect. Comparing against the incorrect derivation can easily mislead evaluation.

Case Study To understand the differences between solution accuracy, equation accuracy and derivation accuracy, we use the following case from the ALG-514 dataset:

Mrs. Martin bought $3(q_1)$ cups of coffee and $2(q_2)$ bagels and spent $12.75(q_3)$ dollars. Mr. Martin bought $2(q_4)$ cups of coffee and $5(q_5)$ bagels and spent $14.00(q_6)$ dollars. Find the cost of one(q_7) cup of coffee and the cost of one(q_8) bagel.

Note that there are six textual numbers (q_1, \dots, q_8) in the word problem. The correct derivation is

$$q_1m + q_2n = q_3$$

$$q_4m + q_5n = q_6.$$

However, without using the derivation annotation, we found that approximation algorithm found the

Supervision	Soln Acc	Eqn Acc	Deriv Acc
ALG-514			
TE	76.2	72.7	75.5
TD	78.4	73.9	77.8
TD - TE	2.2	1.2	2.3
DRAW-1K			
TE	52.0	48.0	48.0
TD	55.0	48.0	53.0
TD - TE	3.0	0	5.0
DOLPHIN			
TE	55.1	50.1	44.2
TD	57.5	36.8	54.9
TD - TE	2.4	-13.3	10.7

Table 3: TE and TD compared using different evaluation metrics. Note that while TD is clearly superior to TE due to extra supervision using the annotations, only derivation accuracy is able to correctly reflect the differences.

wrong derivation and treated it as the correct derivation for this example.

$$q_1m + \mathbf{q_2}n = q_3$$

$$\mathbf{q_2}m + q_5n = q_6.$$

Note while this derivation does generate the correct equation system and solutions, the derivation utilizes the wrong numbers and totally misunderstood the original semantics of the word problem. This example demonstrates the needs to evaluate the quality of the word problem solvers using the annotated derivations.

6 Conclusion and Discussion

In this paper, we propose algorithms for evaluating and annotating derivations for word problems, and introduced a new dataset, DRAW-1K. We also augmented existing benchmarks with derivation annotations to facilitate future comparisons. We found out that derivation accuracy is a more accurate assessment of the ability of solvers for understanding the context of the natural language, compared to solution or equation accuracy.

We also believe that the dataset with the derivation annotations can have new purposes beyond the current intentions. For example, the derivation annotation also allows training systems to produce partial equation system with incomplete word problems. We hope the dataset will open new possibilities for the community to simulate new ideas and applications for automatic problem solvers.

References

- [Bobrow1964] Daniel G. Bobrow. 1964. A question-answering system for high school algebra word problems. In *Proceedings of the October 27-29, 1964, Fall Joint Computer Conference, Part I*, AFIPS '64 (Fall, part I), pages 591–614, New York, NY, USA. ACM.
- [Clark and Etzioni2016] Peter Clark and Oren Etzioni. 2016. My computer is an honor student but how intelligent is it? standardized tests as a measure of ai. *AI Magazine*, 37(1):5–12.
- [Hegarty et al.1995] Mary Hegarty, Richard E. Mayer, and Christopher A. Monk. 1995. Comprehension of arithmetic word problems: A comparison of successful and unsuccessful problem solvers. *Journal Of Educational Psychology*, 87(1):18–32.
- [Hosseini et al.2014] Javad Mohammad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533. Association for Computational Linguistics.
- [Huang et al.2016] Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- [Koncel-Kedziorski et al.2015] Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.
- [Kushman et al.2014] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281. Association for Computational Linguistics.
- [Manning et al.2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.
- [Shi et al.2015] Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1132–1142, Lisbon, Portugal, September. Association for Computational Linguistics.
- [Zhou et al.2015] Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 817–822, Lisbon, Portugal, September. Association for Computational Linguistics.